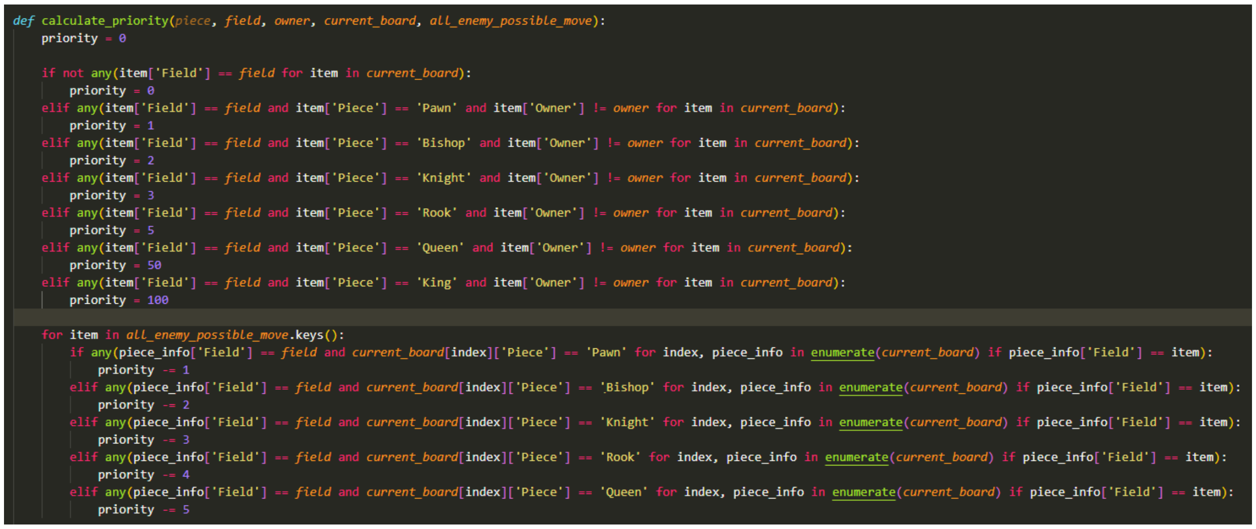


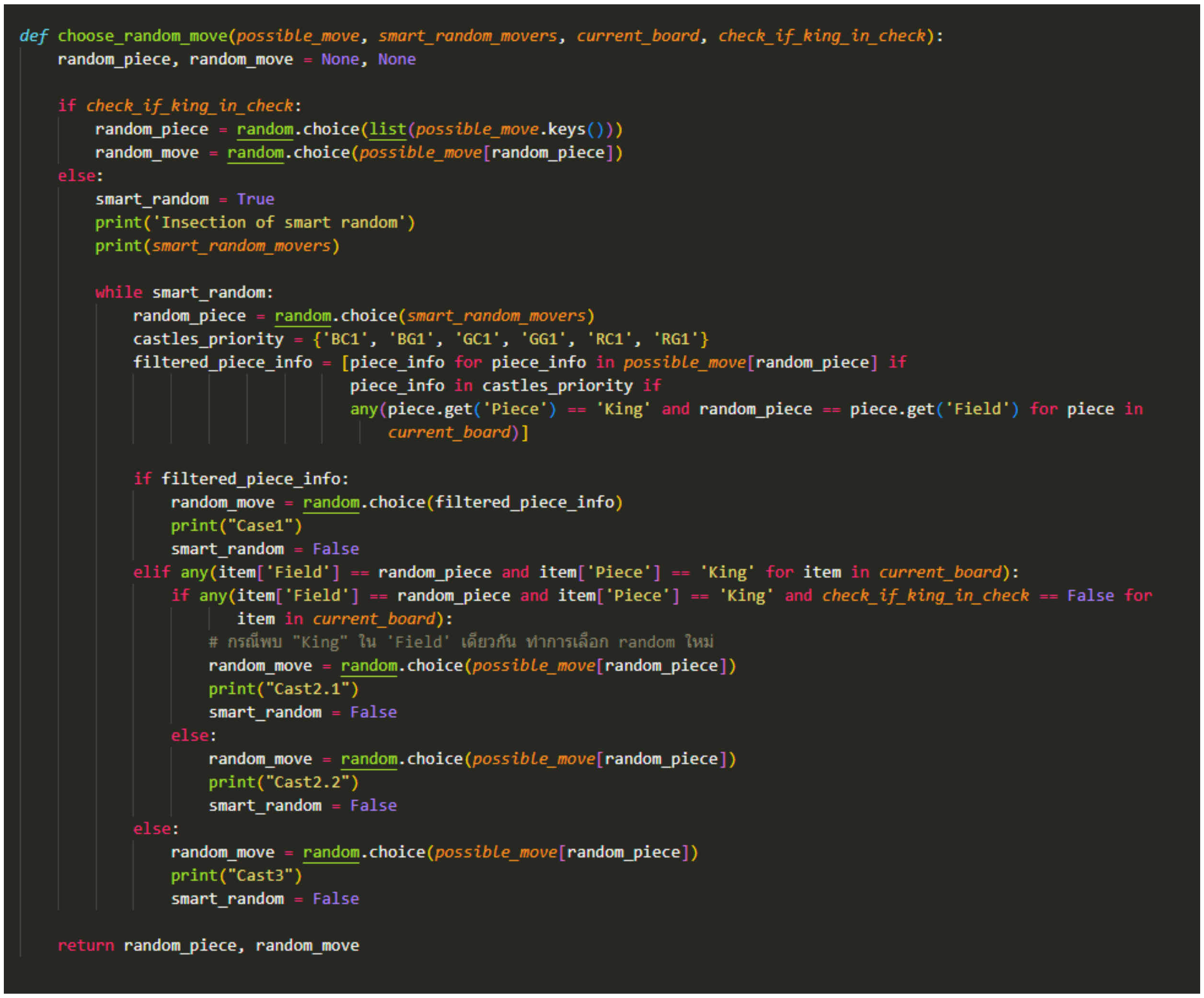
```python
def calculate_priority(piece, field, owner, current_board, all_enemy_possible_move):
    priority = 0

    if not any(item['Field'] == field for item in current_board):
        priority = 0
    elif any(item['Field'] == field and item['Piece'] == 'Pawn' and item['Owner'] != owner for item in current_board):
        priority = 1
    elif any(item['Field'] == field and item['Piece'] == 'Bishop' and item['Owner'] != owner for item in current_board):
        priority = 2
    elif any(item['Field'] == field and item['Piece'] == 'Knight' and item['Owner'] != owner for item in current_board):
        priority = 3
    elif any(item['Field'] == field and item['Piece'] == 'Rook' and item['Owner'] != owner for item in current_board):
        priority = 5
    elif any(item['Field'] == field and item['Piece'] == 'Queen' and item['Owner'] != owner for item in current_board):
        priority = 50
    elif any(item['Field'] == field and item['Piece'] == 'King' and item['Owner'] != owner for item in current_board):
        priority = 100

    for item in all_enemy_possible_move.keys():
        if any(piece_info['Field'] == field and current_board[index]['Piece'] == 'Pawn' for index, piece_info in enumerate(current_board) if piece_info['Field'] == item):
            priority -= 1
        elif any(piece_info['Field'] == field and current_board[index]['Piece'] == 'Bishop' for index, piece_info in enumerate(current_board) if piece_info['Field'] == item):
            priority -= 2
        elif any(piece_info['Field'] == field and current_board[index]['Piece'] == 'Knight' for index, piece_info in enumerate(current_board) if piece_info['Field'] == item):
            priority -= 3
        elif any(piece_info['Field'] == field and current_board[index]['Piece'] == 'Rook' for index, piece_info in enumerate(current_board) if piece_info['Field'] == item):
            priority -= 4
        elif any(piece_info['Field'] == field and current_board[index]['Piece'] == 'Queen' for index, piece_info in enumerate(current_board) if piece_info['Field'] == item):
            priority -= 5
```

calculate_priority()

คำนวณการเดินของหมากว่าช่องไหน priority เท่าไร จะมีค่า priority ตามความใหญ่ของหมากตั้งแต่ Pawn ถึง King

for item in all_enemy_possible_move.keys():

เช็ค move ของผู้เล่นคนอื่นแล้วลบ priority ตามความใหญ่ของหมาก (หมากใหญ่ลบเยอะ)



```python
def choose_random_move(possible_move, smart_random_movers, current_board, check_if_king_in_check):
    random_piece, random_move = None, None

    if check_if_king_in_check:
        random_piece = random.choice(list(possible_move.keys()))
        random_move = random.choice(possible_move[random_piece])
    else:
        smart_random = True
        print('Insection of smart random')
        print(smart_random_movers)

        while smart_random:
            random_piece = random.choice(smart_random_movers)
            castles_priority = {'BC1', 'BG1', 'GC1', 'GG1', 'RC1', 'RG1'}
            filtered_piece_info = [piece_info for piece_info in possible_move[random_piece] if
                                   piece_info in castles_priority if
                                   any(piece.get('Piece') == 'King' and random_piece == piece.get('Field') for piece in
                                       current_board)]

            if filtered_piece_info:
                random_move = random.choice(filtered_piece_info)
                print("Case1")
                smart_random = False
            elif any(item['Field'] == random_piece and item['Piece'] == 'King' for item in current_board):
                if any(item['Field'] == random_piece and item['Piece'] == 'King' and check_if_king_in_check == False for
                       item in current_board):
                    # กรณีพบ "King" ใน 'Field' เดียวกัน ทำการเลือก random ใหม่
                    random_move = random.choice(possible_move[random_piece])
                    print("Cast2.1")
                    smart_random = False
                else:
                    random_move = random.choice(possible_move[random_piece])
                    print("Cast2.2")
                    smart_random = False
            else:
                random_move = random.choice(possible_move[random_piece])
                print("Cast3")
                smart_random = False

    return random_piece, random_move
```

Choose_random_move()

หา possible_move ที่มี priority สูง ถ้าไม่มี priority ที่มากกว่า 0 จะทำการสุ่ม

P.S. ถ้าสุ่มเจอ King และยังไม่โดนเชค แล้วจะสุ่มใหม่ ถ้า King เป็น priority ให้ King เดิน

จะหาหมากที่มี priority มากสุดเดิน โดยการใช้ loop เทียบ priority

ถ้าได้ priority = 0 จะทำการสุ่ม

เนื่องจากเราคำนวณบอร์ดแล้วมาคิดเป็น priority เพื่อดูหากินตัวใหญ่สุด มันก็คือ hill climbing ธรรมดาๆ และไม่ได้คำนวณถึงเทิร์นหน้าๆ เลยทำให้ติดปัญหา local maximum